

FIG. 1A

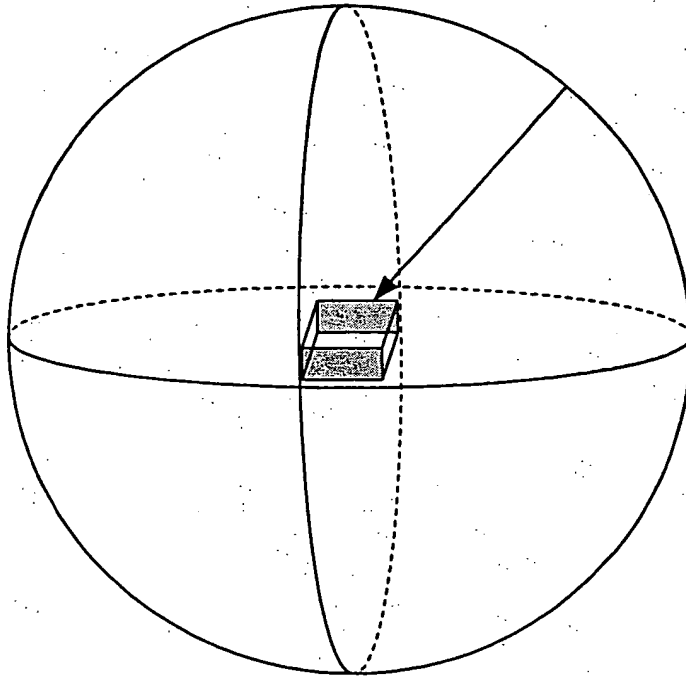
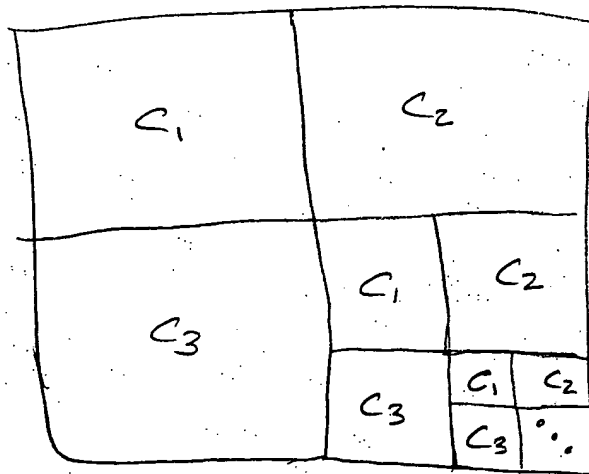


FIG. 1B



$C_1$  = texture 1  
 $C_2$  = texture 2  
 $C_3$  = texture 3

Mip Mapping

FIG. 2

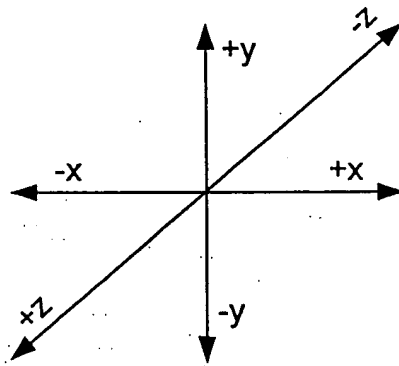
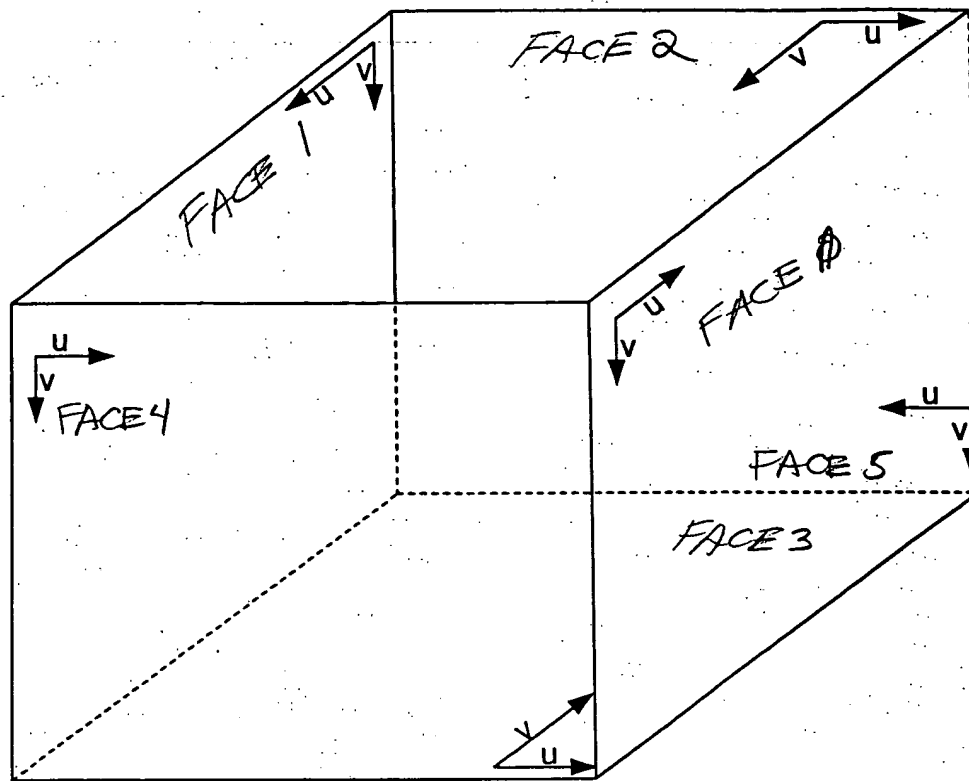


FIG. 3

```
#define FACE_POS_NX 0
#define FACE_NEG_NX 1
#define FACE_POS_NY 2
#define FACE_NEG_NY 3
#define FACE_POS_NZ 4
#define FACE_NEG_NZ 5

float absNx = fabs(Nx);
float absNy = fabs(Ny);
float absNz = fabs(Nz);

if (absNx > absNy && absNx > absNz) { //face major is X or -X
    if (Nx < 0) {
        fid = FACE_NEG_NX;
        U = Nz; V = -Ny; Major = -Nx;
    }
    else {
        fid = FACE_POS_NX;
        U = -Nz; V = -Ny; Major = Nx;
    }
}
else if (absNy > absNz) { //face major is Y or -Y
    if (Ny < 0) {
        fid = FACE_NEG_NY;
        U = Nx; V = -Nz; Major = -Ny;
    }
    else {
        fid = FACE_POS_NY;
        U = Nx; v = Nz; Major = Ny;
    }
}
else { //face major is Z or -Z
    if (Nz < 0) {
        fid = FACE_NEG_NZ;
        U = -Nx; V = -Ny; Major = -Nz;
    }
    else {
        fid = FACE_POS_NZ;
        U = Nx; V = -Ny; Major = Nz;
    }
}
```

FIG. 4

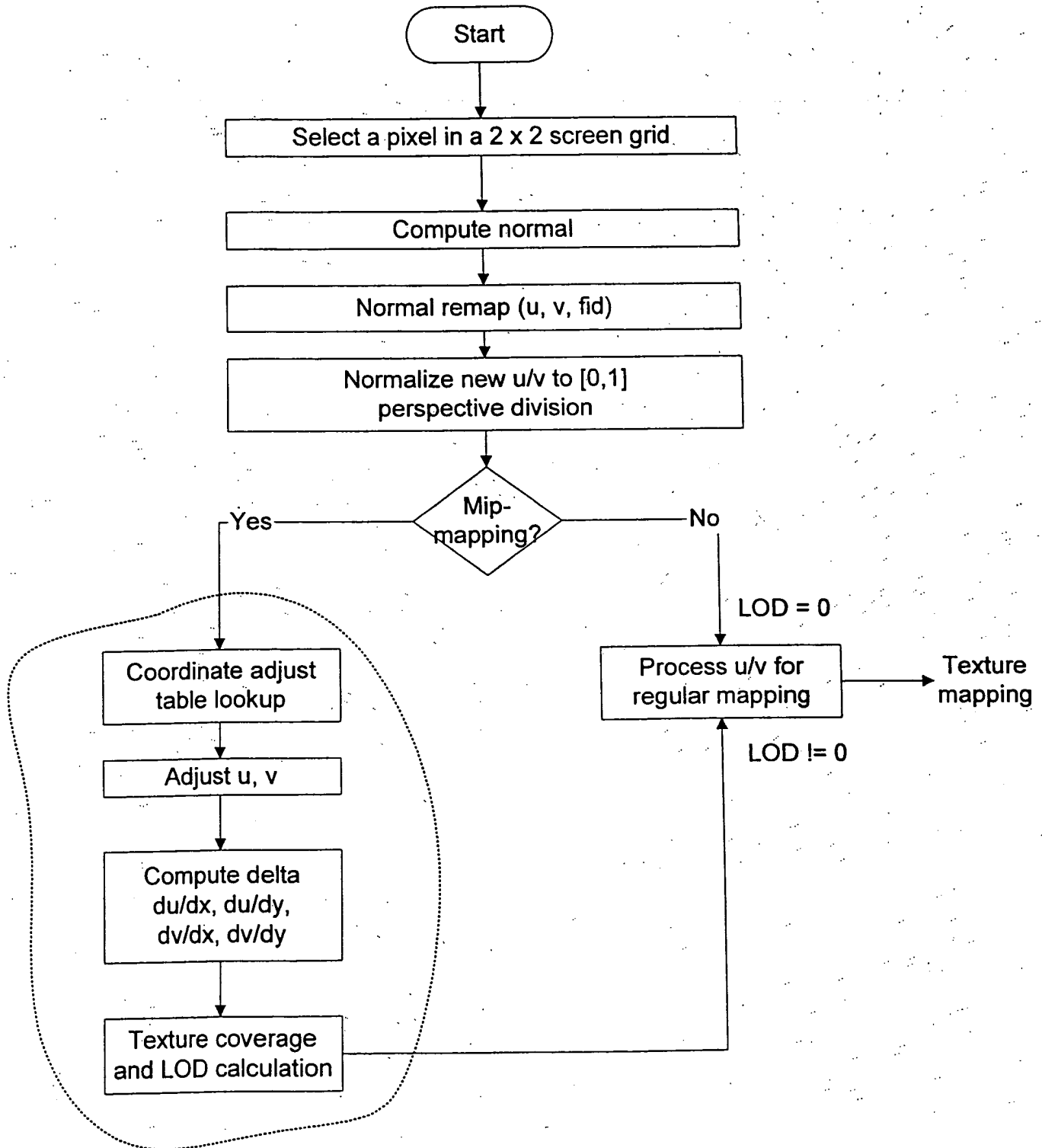


FIG. 5

Bits 3:0						
		0000	1000	1011	1100	1111
Bits 5:4	00	$X \Rightarrow X$ $X \Rightarrow -X$ $-X \Rightarrow X$ $-X \Rightarrow -X$ $Y \Rightarrow Y$ $Y \Rightarrow -Y$ $-Y \Rightarrow Y$ $-Y \Rightarrow -Y$ $Z \Rightarrow Z$ $Z \Rightarrow -Z$ $-Z \Rightarrow Z$ $-Z \Rightarrow -Z$	$X \Rightarrow Z$ $-X \Rightarrow -Z$ $Z \Rightarrow -X$ $-Z \Rightarrow X$	$X \Rightarrow -Y$		$X \Rightarrow Y$
	01		$-Y \Rightarrow Z$ $Z \Rightarrow Y$	$-Y \Rightarrow -X$	$-Y \Rightarrow -Z$ $-Z \Rightarrow -Y$	$-Y \Rightarrow X$
	10		$X \Rightarrow -Z$ $-X \Rightarrow Z$ $Z \Rightarrow X$ $-Z \Rightarrow -X$	$-X \Rightarrow Y$		$-X \Rightarrow -Y$
	11		$Y \Rightarrow Z$ $Z \Rightarrow -Y$	$Y \Rightarrow X$	$Y \Rightarrow -Z$ $-Z \Rightarrow Y$	$Y \Rightarrow -X$

5	4	3	2	1	0
0=add 1=sub	0=U 1=V	Need adding			Swap UV

0	0	No flip
0	1	Flip U
1	0	Flip both UV
1	1	Flip V

FIG. 6

```
uint32 cube_adj_table [6][6] = {  
    // 6 bit code for UV adjustment for x1 during x1-x0  
    // operation. X1's face id and x0's face id are used to  
    // index the table. x1 is either U or V.  
    // bit[0]: swap UV;  
    // bit[2:1]: 10=flip both UV, 00= no flip, 01=flip U,  
    // 11=flip V;  
    // bit[3]: need adding;  
    // bit[5]: 0=add, 1=sub;  
    // bit[4]: 0=U, 1=V;  
    0x00, // X X  
    0x00, // X -X  
    0x0f, // X Y  
    0x0b, // X -Y  
    0x08, // X Z  
    0x28, // X -Z  
  
    0x00, // -X X  
    0x00, // -X -X  
    0x2b, // -X Y  
    0x2f, // -X -Y  
    0x28, // -X Z  
    0x08, // -X -Z  
  
    0x3b, // Y X  
    0x3f, // Y -X  
    0x00, // Y Y  
    0x00, // Y -Y  
    0x38, // Y Z  
    0x3c, // Y -Z  
  
    0x1f, // -Y X  
    0x1b, // -Y -X  
    0x00, // -Y Y  
    0x00, // -Y -Y  
    0x18, // -Y Z  
    0x1c, // -Y -Z  
  
    0x28, // Z X  
    0x08, // Z -X  
    0x18, // Z Y  
    0x38, // Z -Y  
    0x00, // Z Z  
    0x00, // Z -Z  
  
    0x08, // -Z X  
    0x28, // -Z -X  
    0x3c, // -Z Y  
    0x1c, // -Z -Y  
    0x00, // -Z Z  
    0x00, // -Z -Z  
};
```

FIG. 7A



```
uint8 code = cube_adj_table[ fid[1] [fid[0] ];
for (type = 0, 1) repeat { //type 0 is du/dx, type 1 is dv/dx
    bool swap_UV = (code&1);
    bool flip_UV = (( code >> 1&3)==2) ||
                  (( code >> 1&1)      &&
                   type == (code >> 2&1));
    bool add2_UV = !(code >> 5&1) &&
                  (code >> 3&1) &&
                  ((code >> 4&1) == type);
    bool sub2_UV = (code >> 5&1) &&
                  (code >> 3&1) &&
                  ((code >> 4&1) == type);

    if ((swap_UV && type==1) || (!swap_UV && type==0))
        ret = u;
    else
        ret = v;
    if ((flip_UV) { //add one because of u/v already adjusted to 0-1
        ret = 1.0 - ret;
    }
    if (add2_UV) {
        ret += 1.0;
    }
    else if (sub2_UV) {
        ret -= 1.0;
    }
}
}
```

F/G. 7B

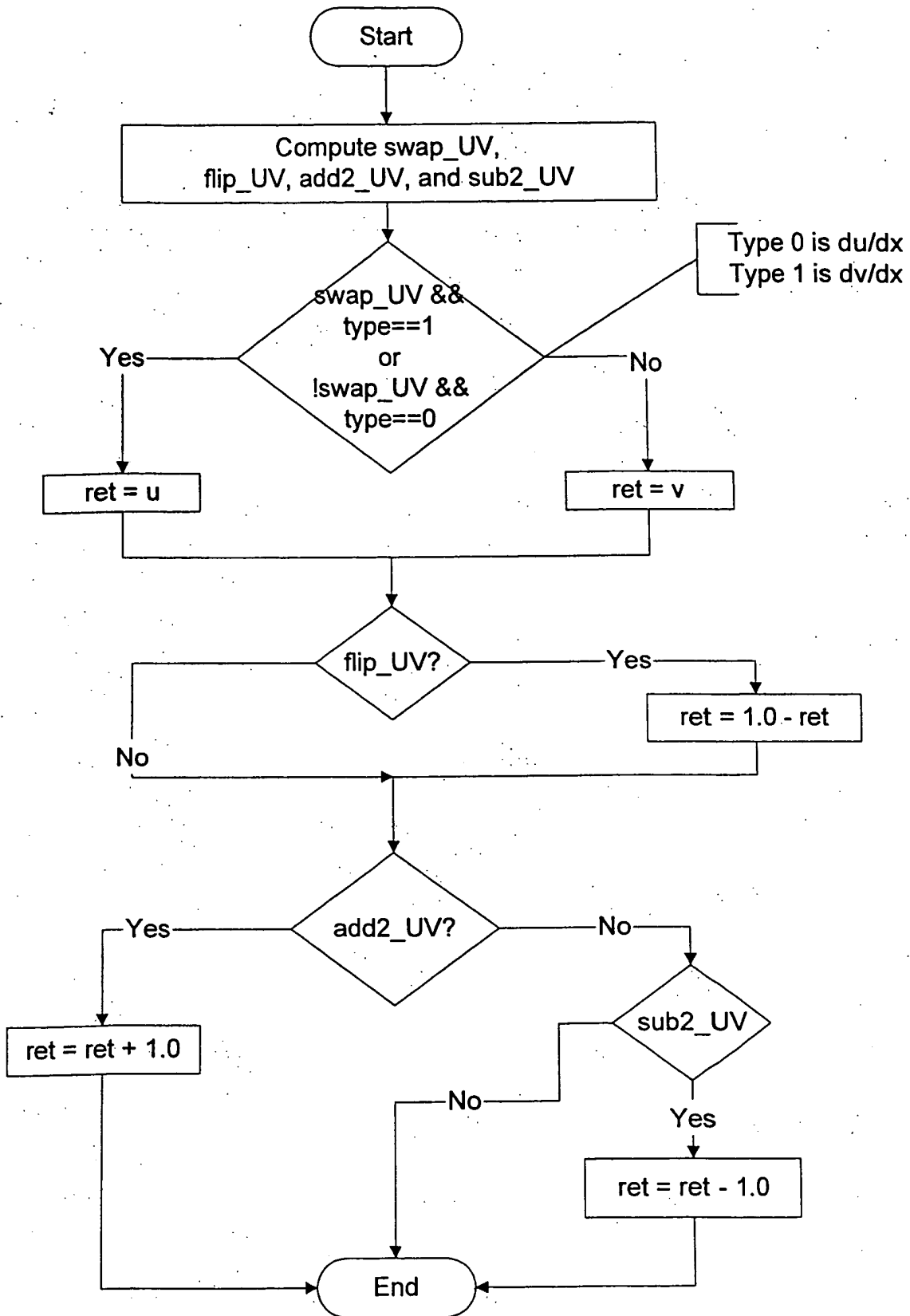


FIG. 8